

Data Formats Work Plan for JetScape

HEPdata <https://hepdata.net/>

Generalized Nuclear Data (GND) <https://ndclx4.bnl.gov/gf/project/gnd/>

Ron Soltz

Nuclear & Particle Physics, LLNL

April 28, 2017



Summary of Thoughts from Last Meeting

- Working assumption: JETSCAPE needs a format that can support co-variance errors
- HEPdata
 - community standard used to store High Energy and Heavy Ion
 - supports statistical and systematic errors, but needs knowledgeable experimentalist to interpret
 - no inherent support for co-variance errors, but technically capable
- Generalized Nuclear Data (GND)
 - only known framework to support co-variance errors
 - supported by BNL and LLNL nuclear data programs
 - python package (fudge) for reading/writing to xml/hdf5
 - current release is fudge.4.21
 - improved support for xml interface in next release

Thoughts on Path Forward

- JETSCAPE must interface with HEPdata (not an option)
- JETSCAPE can either use GND, or we can write our own
- Suppose we begin by testing GND...
 - If we like it, we can continue to use it
 - If we don't, we will write a better one than we would from scratch
- Propose that we write two python interfaces:
 1. Read HEPdata (RAA) and write GND w/ covariant errors
 2. Read GND Co-variance error matrices and write HEPdata table
- Step 1. before analysis, Step 2. when we publish

How this would work

- Python interface to HEPdata goes here
 - Look here for example <https://github.com/HEPData/hepdata-retriever> ?

GND calls to fudge

- [soltz@borax2:gnd]\$ more generateCovariancesGND.py

```
from fudge.gnd.covariances import base
from xData import array, gridded
from xData import axes as axesModule, values as valuesModule, link as linkModule
import numpy

arr = numpy.random.randn( 20,20 ) #random, but could be uniform for full co-variance
# diagonal for independent, or with gaussian or exponential correlation length

GNDarray = array.full( shape=arr.shape, data=arr[ numpy.tril_indices(len(arr)) ], symmetry='lower' )
energyBounds = numpy.logspace(1e-5, 20, 21)
axes = axesModule.axes( labelsUnits = { 0 : ( 'matrix_elements', " " ),
                                         1 : ( 'column_energy_bounds', 'eV' ),
                                         2 : ( 'row_energy_bounds', 'eV' ) } )
axes[2] = axesModule.grid( axes[2].label, axes[2].index, axes[2].unit,
                           style = axesModule.boundariesGridToken, values = valuesModule.values( energyBounds ) )
axes[1] = axesModule.grid( axes[1].label, axes[1].index, axes[1].unit,
                           style = axesModule.linkGridToken, values = linkModule.link( link = axes[2].values, relative = True ) )

GNDgridded = gridded.gridded2d( axes, GNDarray )
covariance = base.covarianceMatrix( "example", type="absolute", matrix=GNDgridded )
print '\n'.join( covariance.toXMMList() )
```

Sample GND output from fudge

- [soltz@borax2:gnd]\$ python generateCovariancesGND.py
- <covarianceMatrix label="example" type="absolute">
- <gridded2d>
- <axes>
- <grid index="2" label="row_energy_bounds" unit="eV" style="boundaries">
- <values length="21">1.00002302611603 10.0002187479763 100.002072348057 1000.01957216482 10000.1842085041 100001.726953731
10000016.11822555 10000149.6691511 100001381.560599 1000012664.2982 10000115129.9174 100001036168.66 1000009210382.79 10000080590803
100000690777914 1000005756479303 10000046051807946 1.0000034538836e17 1.00000230258775e18 1.00000115129321e19 1e20</values></grid>
- <grid index="1" label="column_energy_bounds" unit="eV" style="link">
- <link xlink:href="..../grid[@index='2']/values"/></grid>
- <axis index="0" label="matrix_elements" unit=""></axis>
- <array shape="20,20" symmetry="lower">
- <values length="210">0.606749694302289 -2.29686531124636 1.14040777223007 1.09727822296894 -0.437361751140772 -4.97777683079598
1.29558675246568 -1.97436433699583 0.324973354339331 3.17789672111136 -2.55231739370079 1.64394493684898 -0.288027360665713 -
0.362483264547509 -1.21012305800322 9.91163418124774e-3 -2.64837653546381 -2.21755591966909 -0.0764956193148251 -1.057682206350599
2.47189663393128 -1.82927622371598 -0.0866504749365749 0.615428493634386 0.0380303649224523 -0.0181042868409647 -2.34593525195783 -
1.02156715299238 1.73194377936738 0.224738163900254 -0.226537660226752 -0.397341381109827 2.45529627592146 3.62470583665945 -
1.19322734132332 -1.03452000697876 2.97823846647251 -2.24043141841074 0.371288255012666 -1.96498930088126 0.544924607906588 -
1.83724123412784 0.670135443190461 0.501931172883976 -0.640379802959672 -2.00560572845017 -1.25474067983429 0.337711160642448 -
1.01220343820835 1.6130919405377 -1.58161489420604 -0.411840983368557 0.616009848587437 1.85695124148109 1.68493832502138 -
2.03085324442969 -2.60949416806441 -0.560418322318083 -1.30409435759143 -1.08452216851094 -0.658725975467499 -0.474899994800445 ...

Work Plan

- Start work end of May, (new fudge release expected)
- Need $\frac{1}{4}$ - $\frac{1}{2}$ experimentalist to work with LLNL summer student to read R_{AA} from HEPdata and fill error matrices appropriately
- Also needed, statistician to incorporate co-variance errors into Bayesian optimization for R_{AA}
- Plan for test case to be ready by collaboration meeting